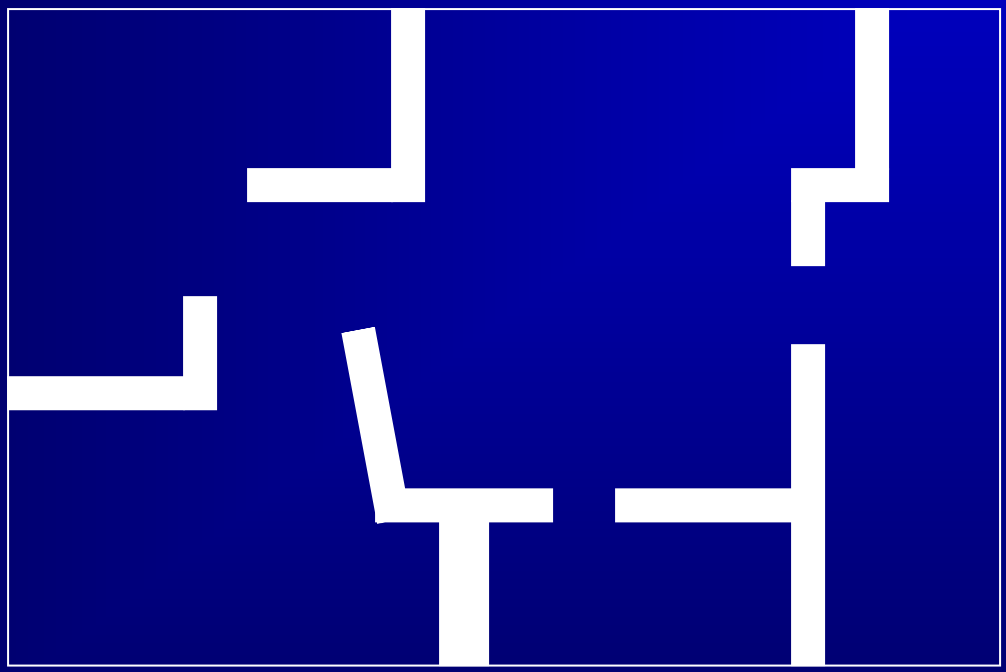
Guarding and Triangulation

Outline

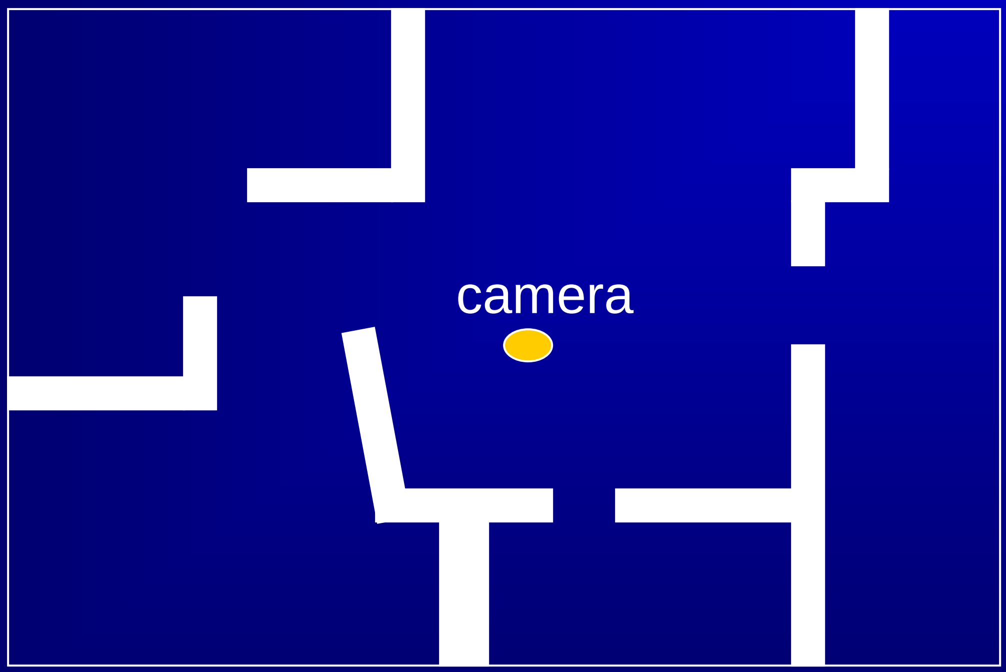
I. Properties of polygon triangulation

II. Solution to the art gallery probelm

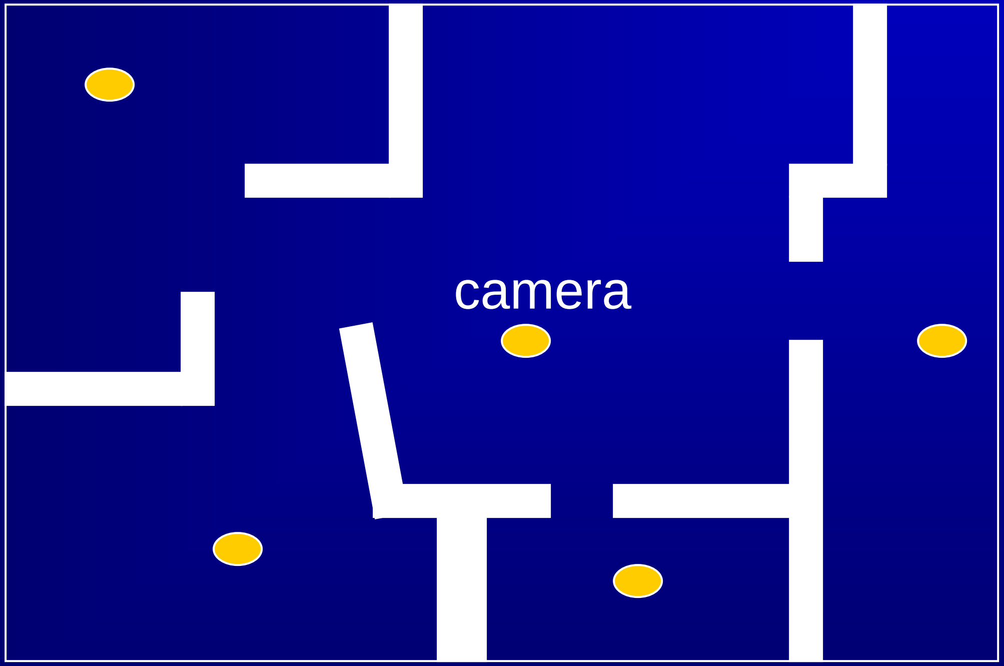
The Art Gallery Problem



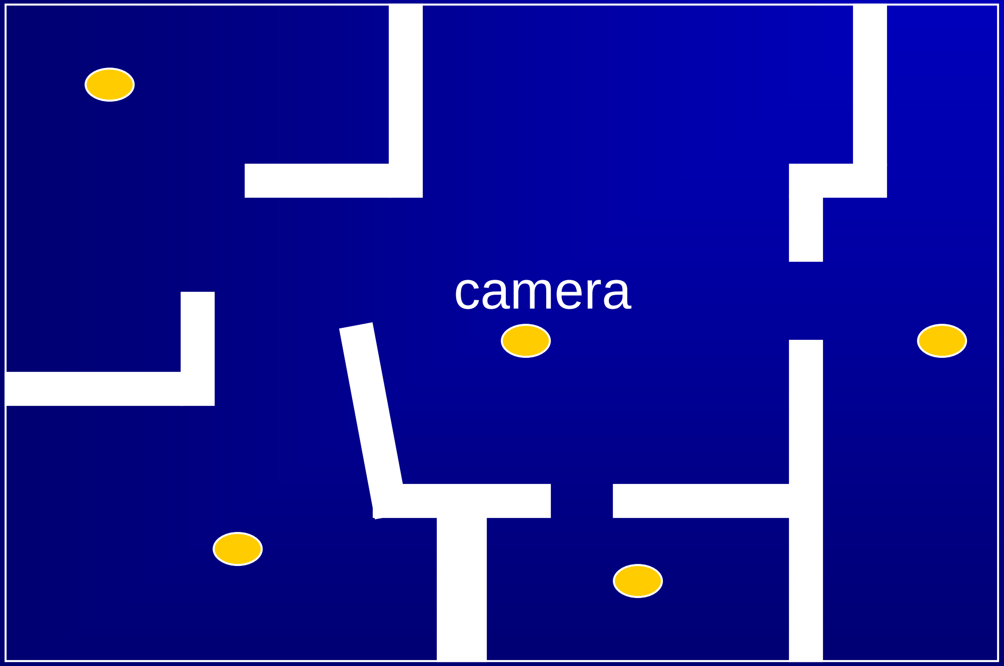
The Art Gallery Problem



The Art Gallery Problem



The Art Gallery Problem



How many cameras are needed to guard a gallery?
Where should they be placed?

I. Simple Polygon Model

Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

I. Simple Polygon Model

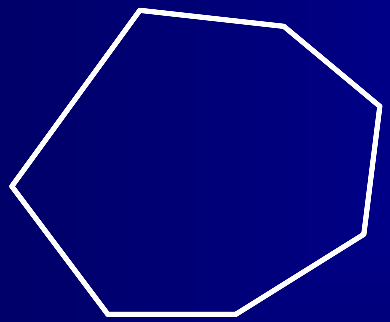
Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

Regions with holes are not allowed.

I. Simple Polygon Model

Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

Regions with holes are not allowed.

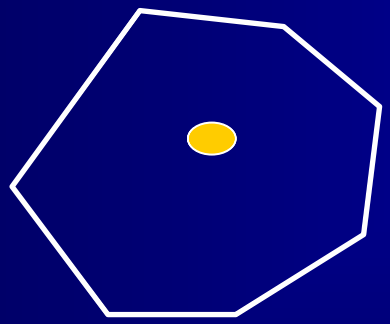


convex polygon

I. Simple Polygon Model

Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

Regions with holes are not allowed.

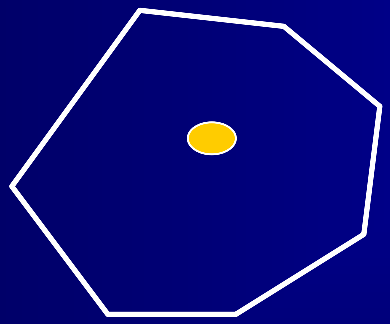


convex polygon

I. Simple Polygon Model

Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

Regions with holes are not allowed.

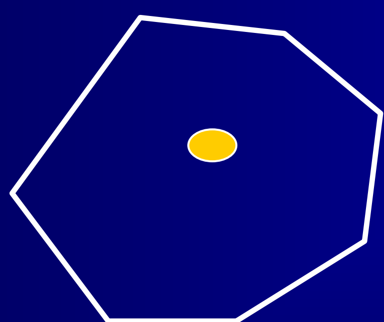


convex polygon
one camera

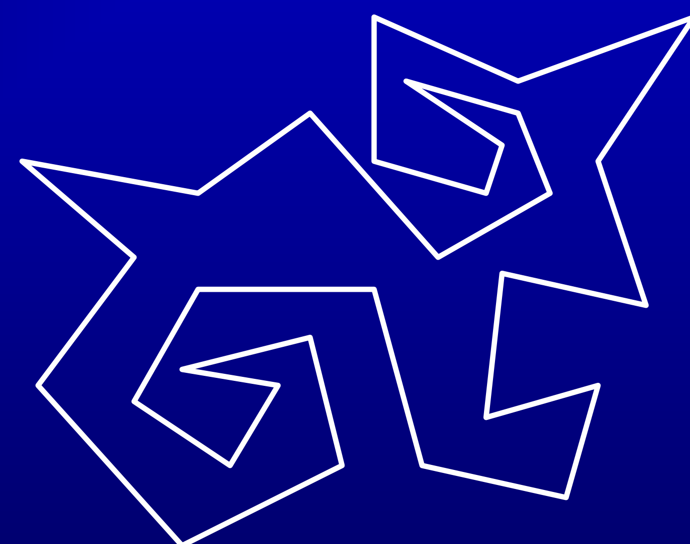
I. Simple Polygon Model

Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

Regions with holes are not allowed.



convex polygon
one camera

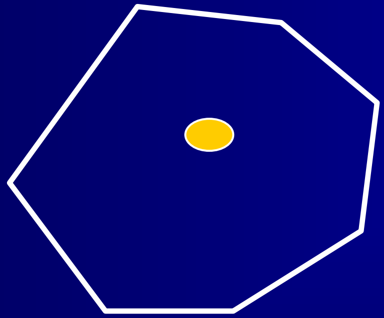


an arbitrary n-gon (n vertices)

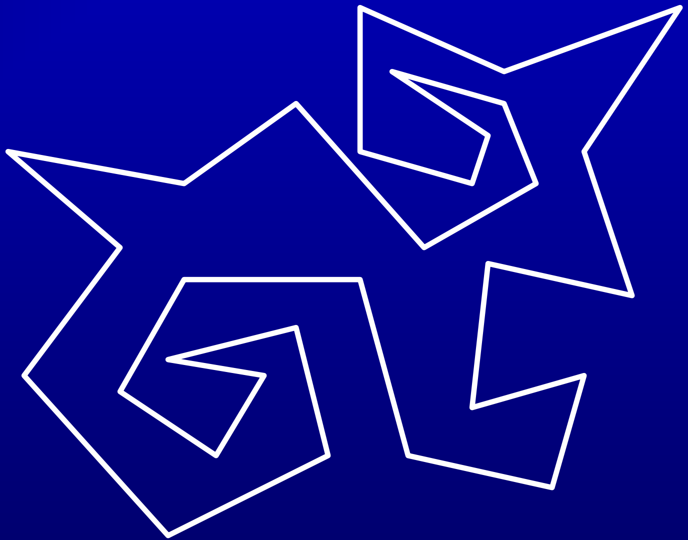
I. Simple Polygon Model

Model the art gallery as a region bounded by some *simple polygon* (no self-crossing).

Regions with holes are not allowed.



convex polygon
one camera

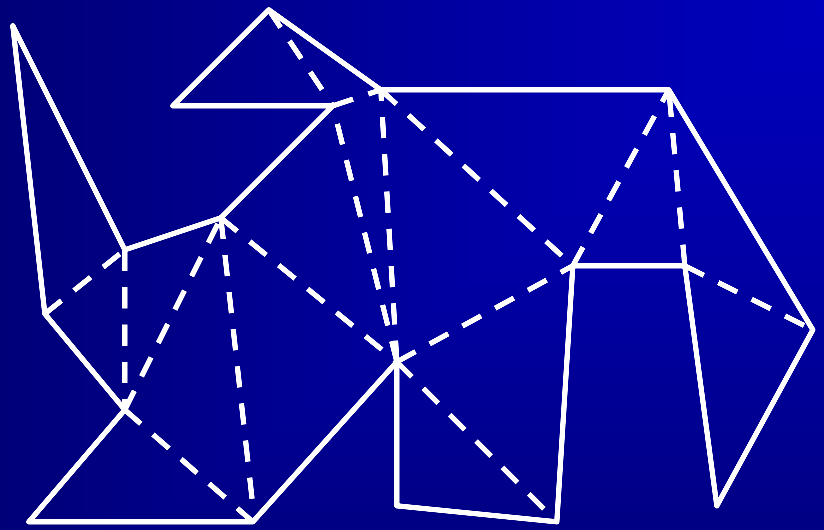


an arbitrary n-gon (n vertices)

Bad news: finding the minimum number of cameras for a given polygon is NP-hard (exponential time).

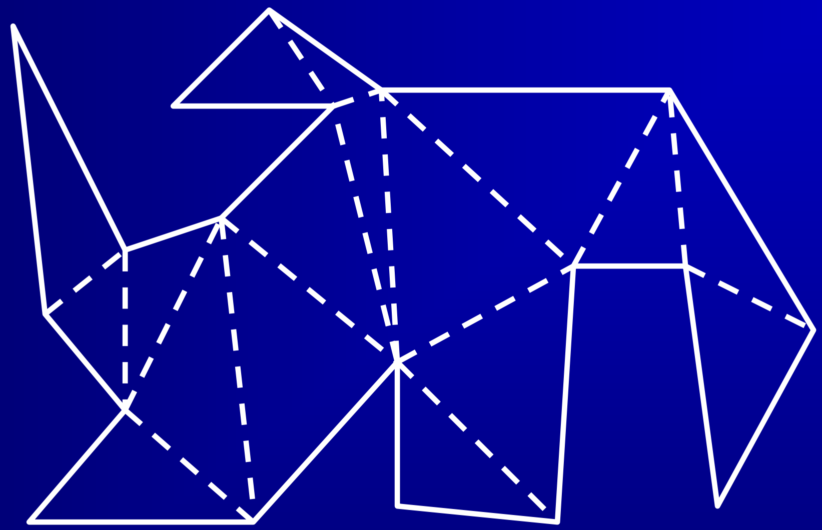
Triangulation

To make things easier, we decompose a polygon into pieces that are easy to guard.



Triangulation

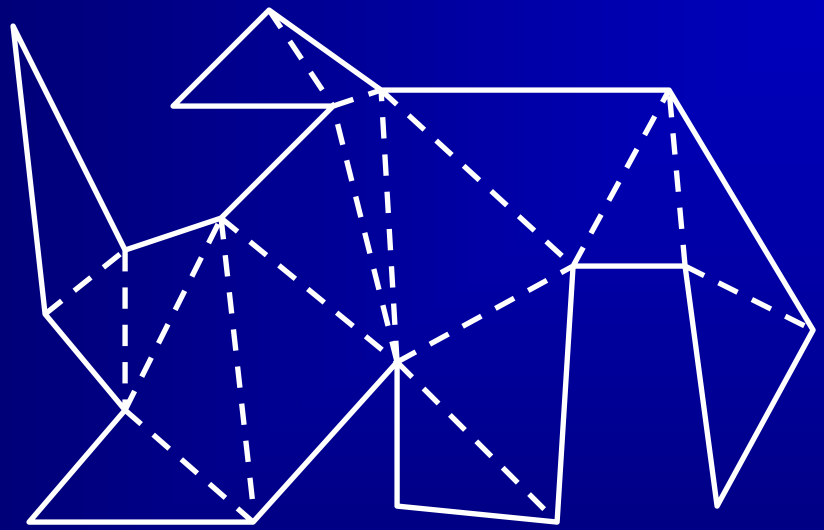
To make things easier, we decompose a polygon into pieces that are easy to guard.



Draw *diagonals* between pair of vertices.

Triangulation

To make things easier, we decompose a polygon into pieces that are easy to guard.

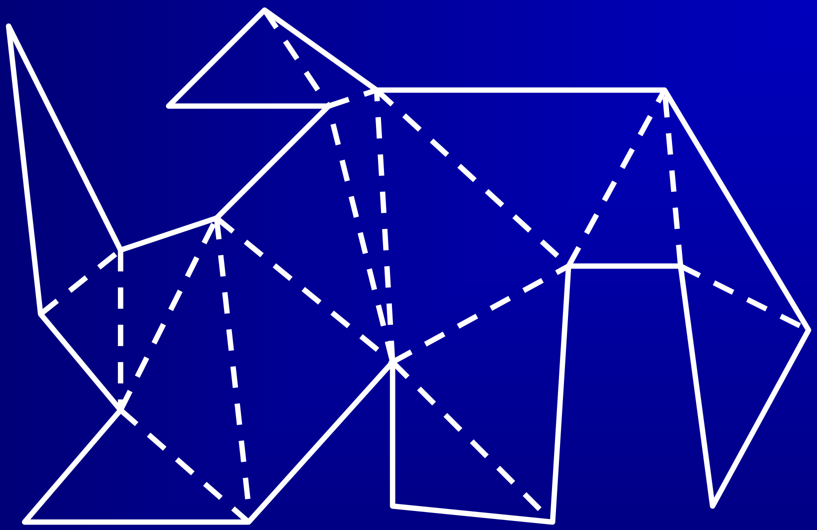


Draw *diagonals* between pair of vertices.

an open line segment that connects two vertices and lie in the interior of the polygon.

Triangulation

To make things easier, we decompose a polygon into pieces that are easy to guard.



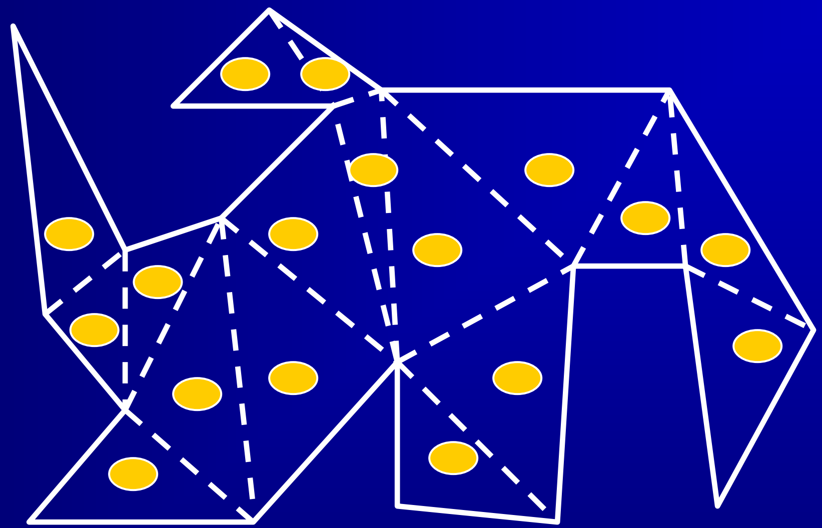
Draw *diagonals* between pair of vertices.

an open line segment that connects two vertices and lie in the interior of the polygon.

Guard the polygon by placing a camera in every triangle …

Triangulation

To make things easier, we decompose a polygon into pieces that are easy to guard.



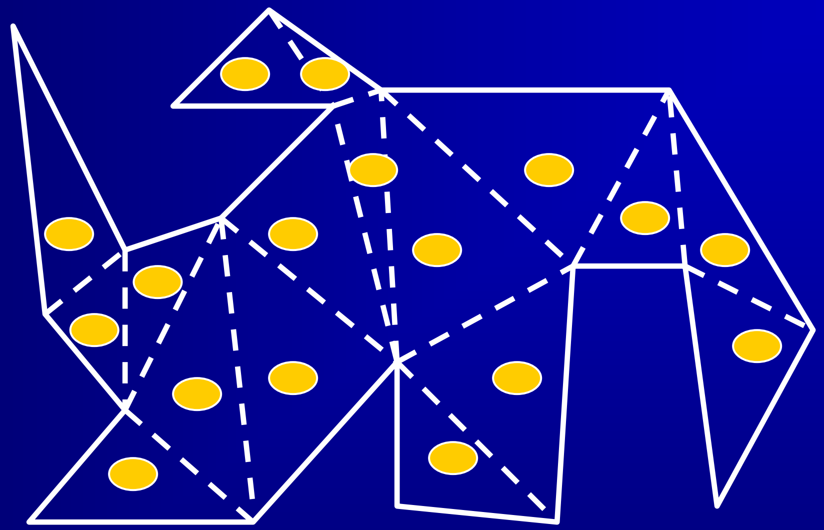
Draw *diagonals* between pair of vertices.

an open line segment that connects two vertices and lie in the interior of the polygon.

Guard the polygon by placing a camera in every triangle …

Triangulation

To make things easier, we decompose a polygon into pieces that are easy to guard.



Draw *diagonals* between pair of vertices.

an open line segment that connects two vertices and lie in the interior of the polygon.

Guard the polygon by placing a camera in every triangle …

Triangulation: decomposition of a polygon into triangles by a maximal set of non-intersecting diagonals.

Triangles

Theorem 1 Every simple polygon has a triangulation. Any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

Triangles

Theorem 1 Every simple polygon has a triangulation. Any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

Proof By induction.

Triangles

Theorem 1 Every simple polygon has a triangulation.
Any triangulation of a simple polygon with n vertices consists of exactly $n - 2$ triangles.

Proof By induction.

Trivial for $n = 3$.

Triangles

Theorem 1 Every simple polygon has a triangulation.
Any triangulation of a simple polygon with n vertices consists of exactly $n - 2$ triangles.

Proof By induction.

Trivial for $n = 3$.

Assume true for all $m < n$.

Triangles

Theorem 1 Every simple polygon has a triangulation.
Any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

Proof By induction.

Trivial for $n = 3$.

Assume true for all $m < n$.

- Existence

Triangles

Theorem 1 Every simple polygon has a triangulation. Any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

Proof By induction.

Trivial for $n = 3$.

Assume true for all $m < n$.

- Existence

  Let v be the leftmost vertex and u and w its two neighbors.

Triangles

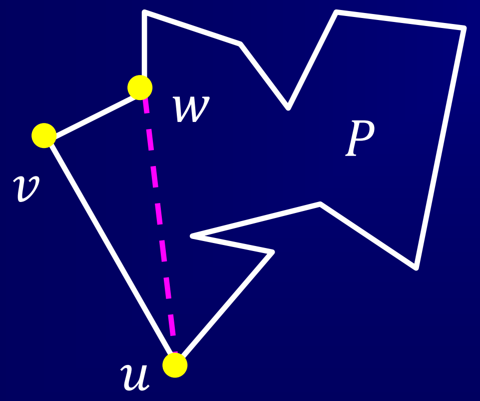
Theorem 1 Every simple polygon has a triangulation.
Any triangulation of a simple polygon with n vertices consists of exactly $n - 2$ triangles.

Proof By induction.

Trivial for $n = 3$.

Assume true for all $m < n$.

- Existence



Let v be the leftmost vertex and u and w its two neighbors.

- $\overline{uw}$ in the interior of P $\Rightarrow$ it is a diagonal.

Triangles

Theorem 1 Every simple polygon has a triangulation.
Any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

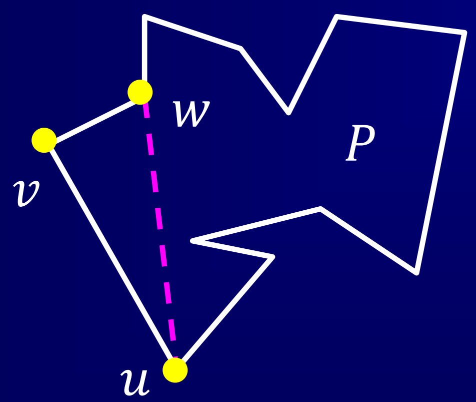
Proof By induction.

Trivial for $n = 3$.

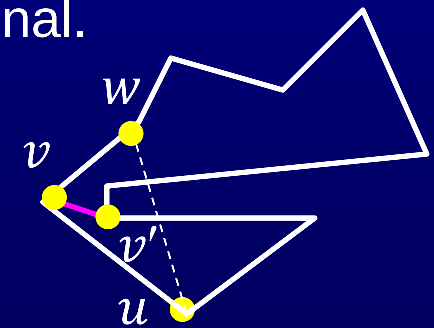
Assume true for all $m < n$.

- Existence

Let v be the leftmost vertex and u and w its two neighbors.



- $\overline{uw}$ in the interior of P $\Rightarrow$ it is a diagonal.
- Otherwise, the triangle determined by u, v, w contains at least one vertex. Let v' be the one closest to v. Then $\overline{vv'}$ is a diagonal.



Triangles

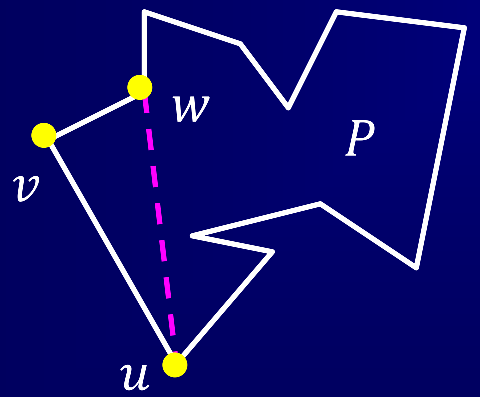
Theorem 1 Every simple polygon has a triangulation.
Any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

Proof By induction.

Trivial for $n = 3$.

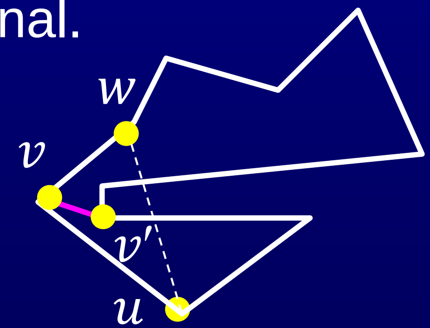
Assume true for all $m < n$.

- Existence



Let v be the leftmost vertex and u and w its two neighbors.

- $\overline{uw}$ in the interior of P $\Rightarrow$ it is a diagonal.
- Otherwise, the triangle determined by u, v, w contains at least one vertex. Let v' be the one closest to v. Then $\overline{vv'}$ is a diagonal.



The diagonal splits the polygon into two (which by induction can be triangulated).

Proof (cont'd)

Proof (cont'd)

- # triangles = $n - 2$

Proof (cont'd)

- # triangles = $n - 2$

  Any diagonal splits P into two simple polygons with k and m vertices, respectively.

Proof (cont'd)

- # triangles $= n - 2$

  Any diagonal splits P into two simple polygons with k and m vertices, respectively.

  By induction these two subpolygons can be triangulated.

Proof (cont'd)

- # triangles = $n - 2$

Any diagonal splits P into two simple polygons with k and m vertices, respectively.

By induction these two subpolygons can be triangulated.

They are decomposed into $k-2$ and $m-2$ triangles, resp.

Proof (cont'd)

- # triangles = $n-2$

  Any diagonal splits P into two simple polygons with k and m vertices, respectively.

  By induction these two subpolygons can be triangulated.

  They are decomposed into $k-2$ and $m-2$ triangles, resp.

  - Vertices defining the diagonal occur in each subpolygon once. These two vertices contribute 2 each to k and m.
  - Other vertices of P each occurs in exactly on one subpolygon.

Proof (cont'd)

- # triangles $= n - 2$

  Any diagonal splits P into two simple polygons with k and m vertices, respectively.

  By induction these two subpolygons can be triangulated.

  They are decomposed into $k-2$ and $m-2$ triangles, resp.

  - Vertices defining the diagonal occur in each subpolygon once. These two vertices contribute 2 each to k and m.
  - Other vertices of P each occurs in exactly on one subpolygon.

  Thus $k + m = n + 2$.

Proof (cont'd)

- # triangles = $n-2$

Any diagonal splits P into two simple polygons with k and m vertices, respectively.

By induction these two subpolygons can be triangulated.

They are decomposed into $k-2$ and $m-2$ triangles, resp.

- Vertices defining the diagonal occur in each subpolygon once. These two vertices contribute 2 each to k and m.
- Other vertices of P each occurs in exactly on one subpolygon.

Thus $k+m=n+2$.

By induction, the triangulation of P has
$(k-2)+(m-2)=k+m-4=n+2-4=n-2$
triangles.

Proof (cont'd)

- # triangles = $n - 2$

  Any diagonal splits P into two simple polygons with k and m vertices, respectively.

  By induction these two subpolygons can be triangulated.

  They are decomposed into $k - 2$ and $m - 2$ triangles, resp.

  - Vertices defining the diagonal occur in each subpolygon once. These two vertices contribute 2 each to k and m.

  - Other vertices of P each occurs in exactly on one subpolygon.

  Thus $k + m = n + 2$.

  By induction, the triangulation of P has
  $(k - 2) + (m - 2) = k + m - 4 = n + 2 - 4 = n - 2$
  triangles.

II. # Cameras for AG

Theorem 1 $\Rightarrow$ $n - 2$ cameras can guard the simple polygon.

II. # Cameras for AG

Theorem 1 $\Rightarrow$ $n-2$ cameras can guard the simple polygon.

A camera on a diagonal guards two triangles.

II. # Cameras for AG

Theorem 1 $\Rightarrow$ $n - 2$ cameras can guard the simple polygon.

A camera on a diagonal guards two triangles.

$\Rightarrow$ # cameras can be reduced to roughly $n/2$.

II. # Cameras for AG

Theorem 1 $\Rightarrow$ $n-2$ cameras can guard the simple polygon.

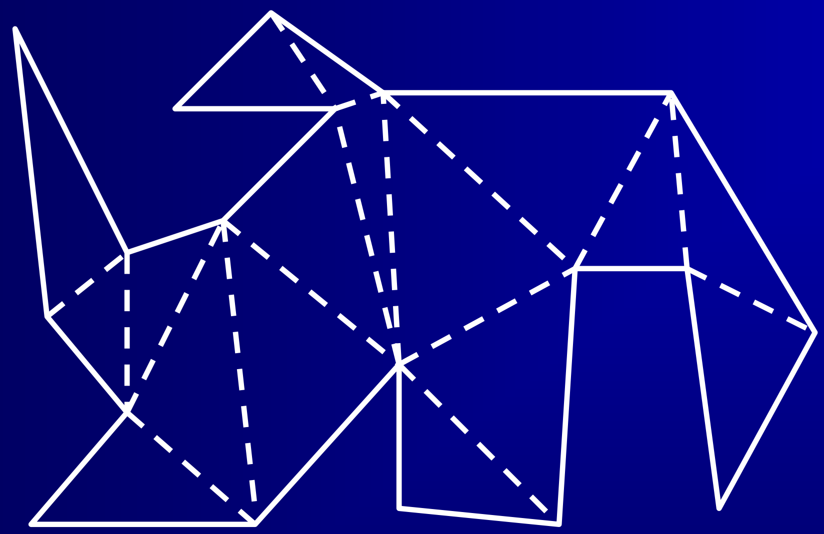
A camera on a diagonal guards two triangles.

$\Rightarrow$ # cameras can be reduced to roughly $n/2$.

A vertex is adjacent to many triangles.
So placing cameras at vertices can do even better …

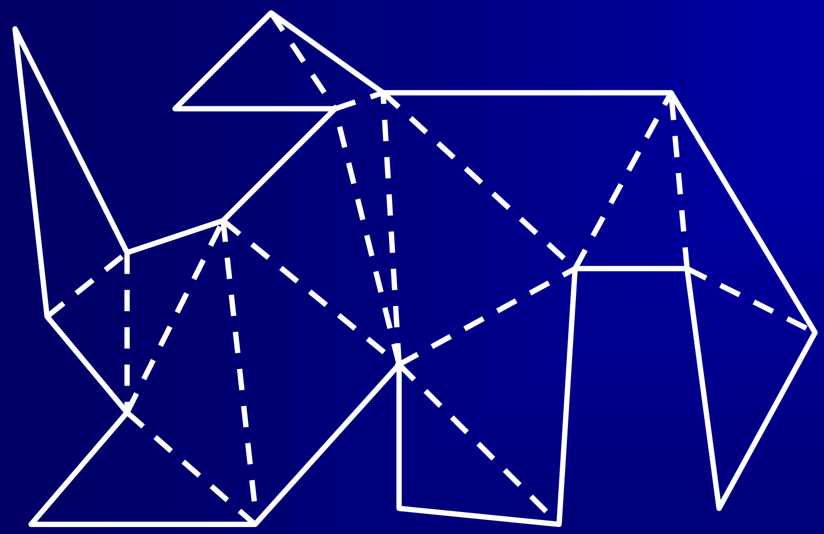
3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.



3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.

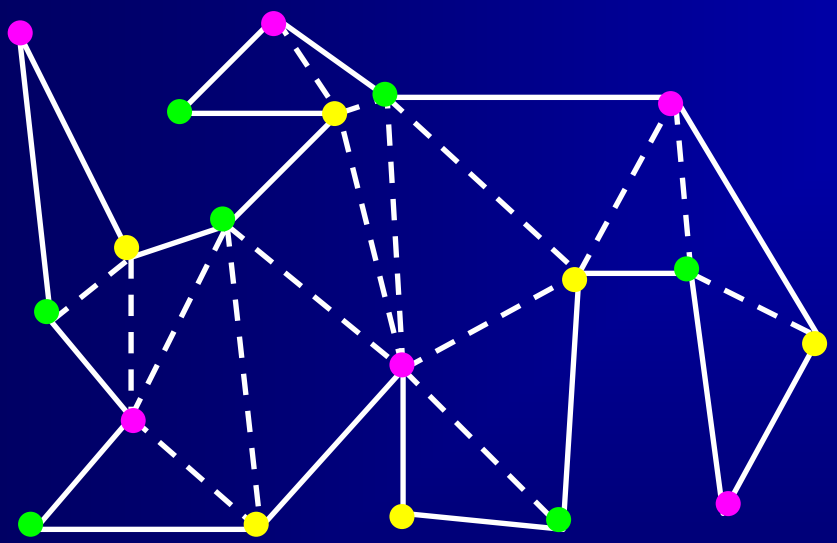


A 3-coloring exists if

- every vertex is assigned a color: pink, green, or yellow, and

3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.



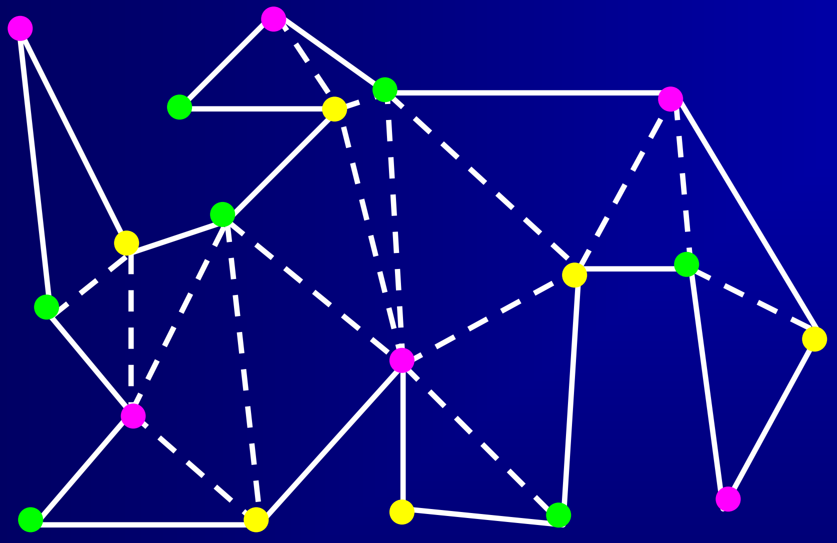
A 3-coloring exists if

- every vertex is assigned a color: pink, green, or yellow, and

3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.

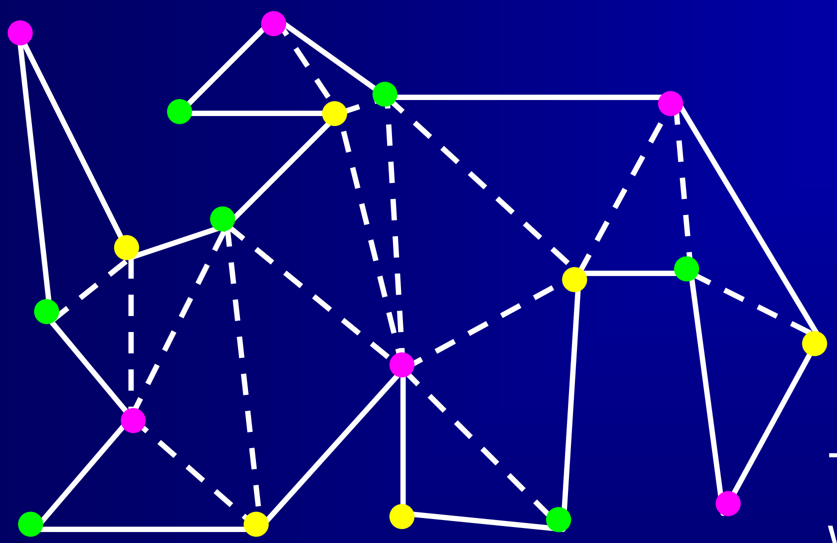
A 3-coloring exists if



- every vertex is assigned a color: pink, green, or yellow, and
- any two vertices connected by an edge or a diagonal are assigned different colors.

3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.



A 3-coloring exists if

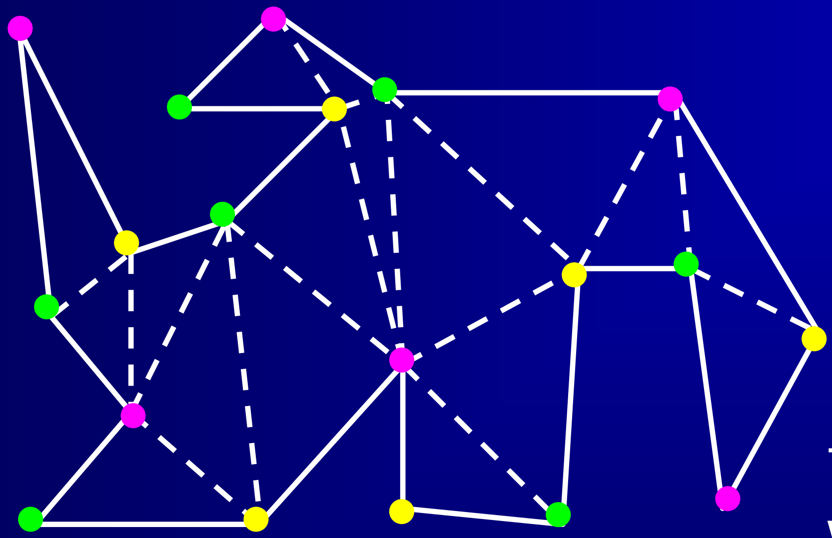
- every vertex is assigned a color: pink, green, or yellow, and
- any two vertices connected by an edge or a diagonal are assigned different colors.

Thus, the vertices of every triangle will be in three different colors.

3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.

A 3-coloring exists if



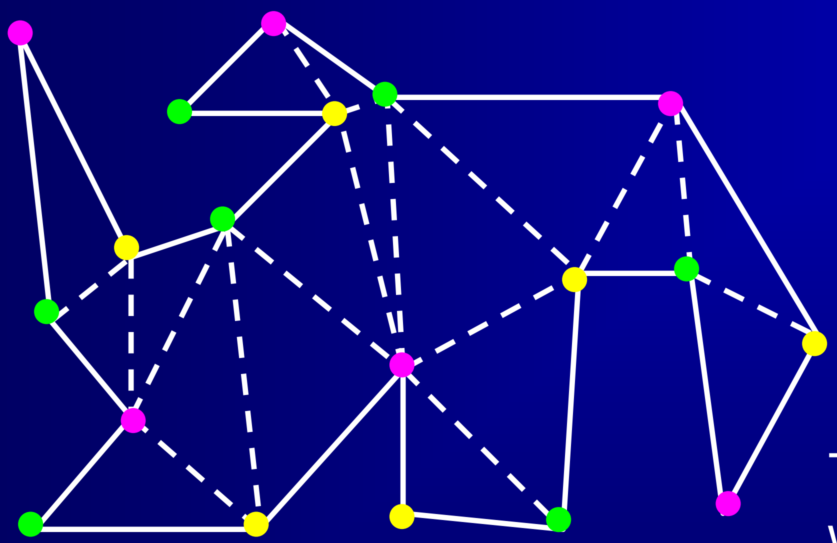
- every vertex is assigned a color: pink, green, or yellow, and
- any two vertices connected by an edge or a diagonal are assigned different colors.

Thus, the vertices of every triangle will be in three different colors.

If 3-coloring exists, place cameras at all vertices of the same color.

3-Coloring

Idea: Select a set of vertices, such that any triangle has at least one selected vertex and place cameras at selected vertices.



A 3-coloring exists if

- every vertex is assigned a color: pink, green, or yellow, and
- any two vertices connected by an edge or a diagonal are assigned different colors.

Thus, the vertices of every triangle will be in three different colors.

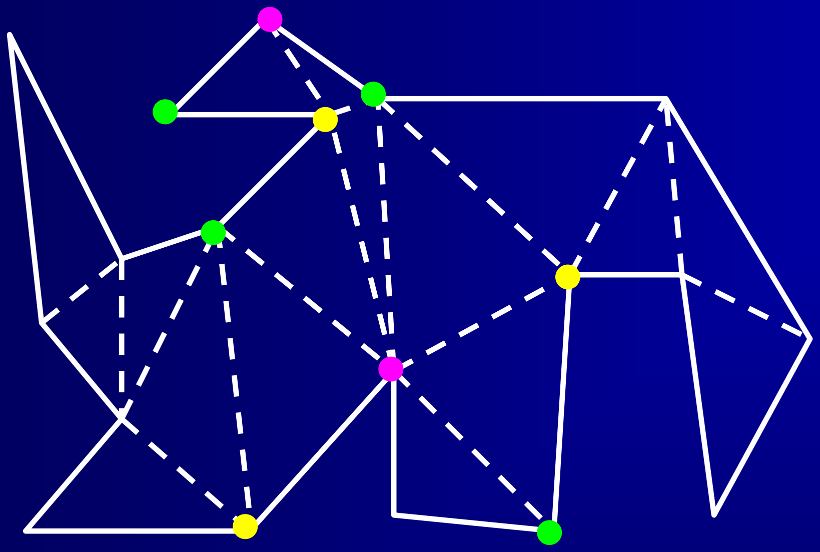
If 3-coloring exists, place cameras at all vertices of the same color.

Choose the smallest color class to place the cameras.

$\Rightarrow \lfloor n/3 \rfloor$ cameras.

The Dual Graph

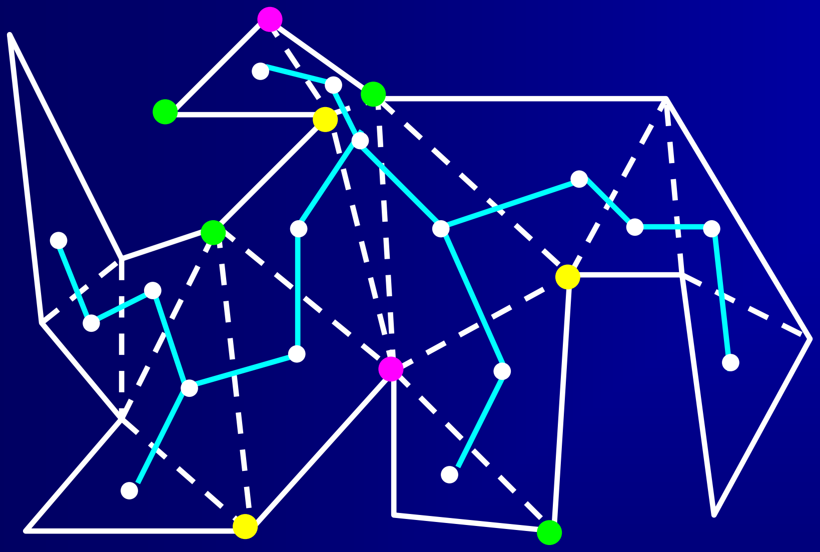
Dual graph G has a node inside every triangle and an edge between every pair of nodes whose corresponding triangles share a diagonal.



$\Longrightarrow$

The Dual Graph

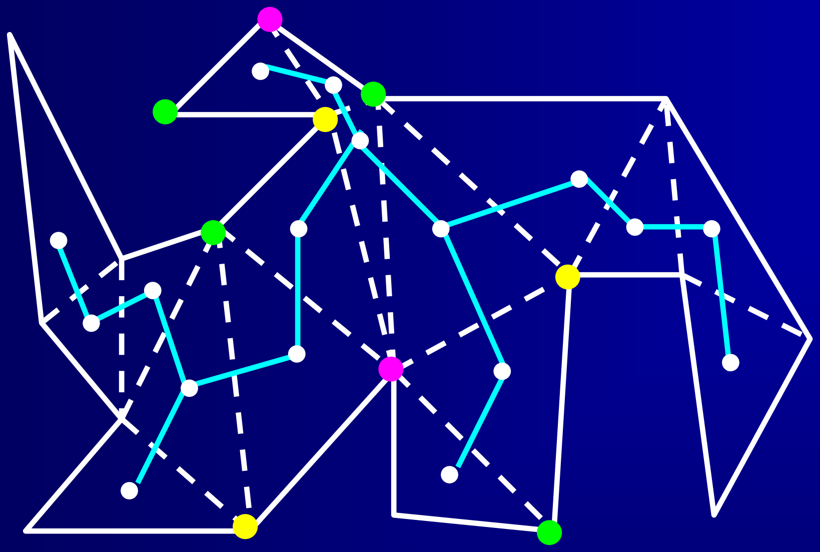
Dual graph G has a node inside every triangle and an edge between every pair of nodes whose corresponding triangles share a diagonal.



$\Rightarrow$

The Dual Graph

Dual graph G has a node inside every triangle and an edge between every pair of nodes whose corresponding triangles share a diagonal.

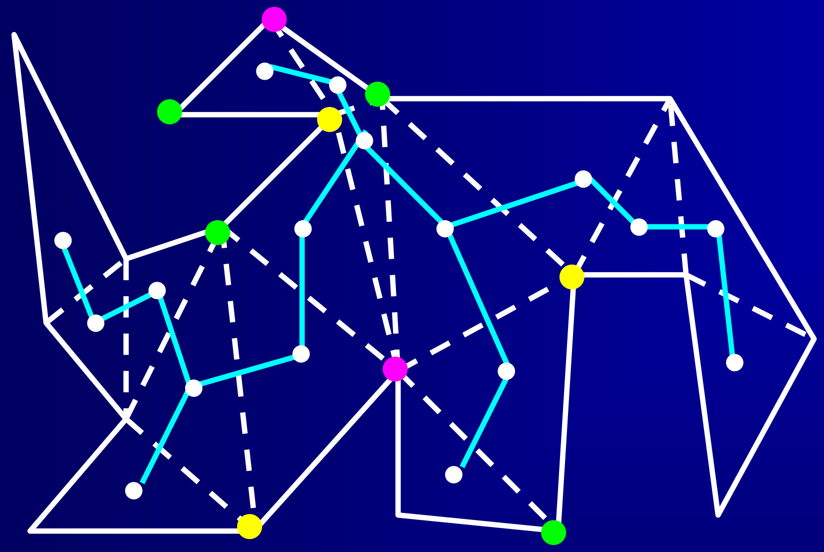


G is connected.

$\Rightarrow$

The Dual Graph

Dual graph G has a node inside every triangle and an edge between every pair of nodes whose corresponding triangles share a diagonal.



G is connected.

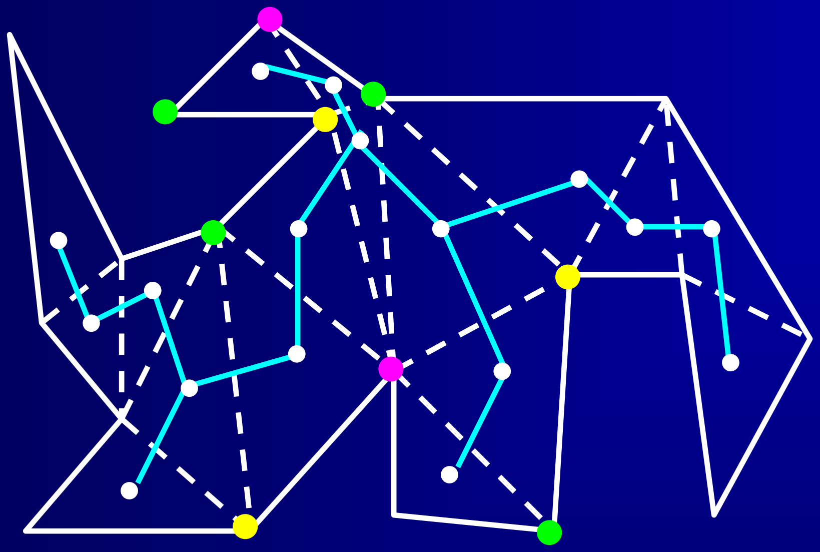
Any diagonal cuts the polygon into two.

Every diagonal corresponds to an edge in the dual graph.

Removal of any edge from the dual graph disconnects it.

The Dual Graph

Dual graph G has a node inside every triangle and an edge between every pair of nodes whose corresponding triangles share a diagonal.



G is connected.

Any diagonal cuts the polygon into two.

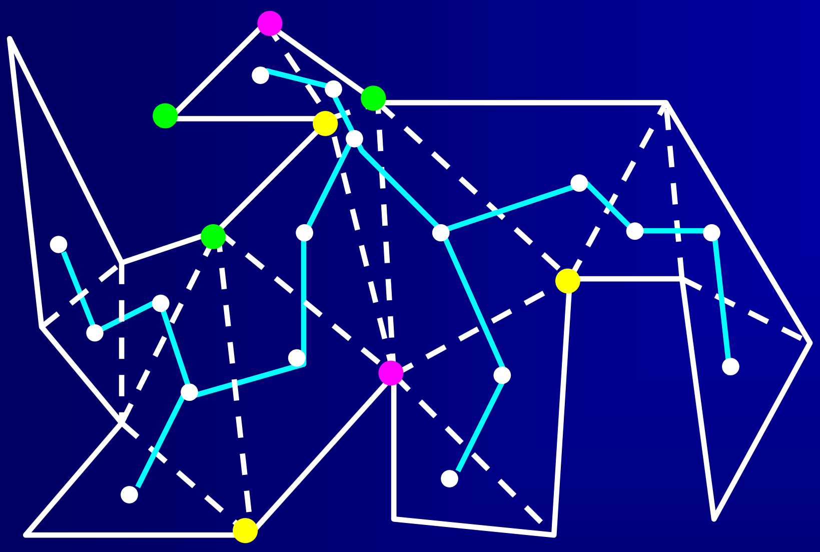
Every diagonal corresponds to an edge in the dual graph.

Removal of any edge from the dual graph disconnects it.

Thus, the dual graph is a tree.

A 3-Coloring Algorithm

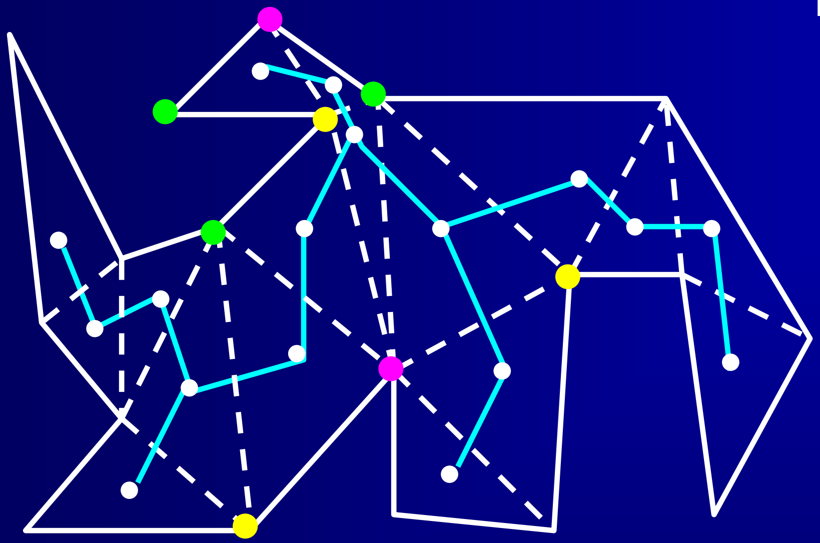
A 3-coloring can be found through a graph traversal (such as DFS).



$\Rightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



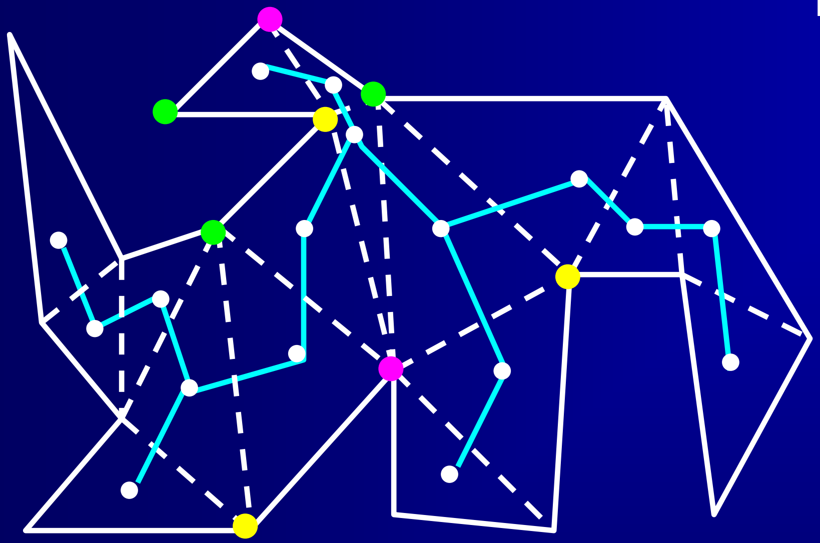
During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

$\Rightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

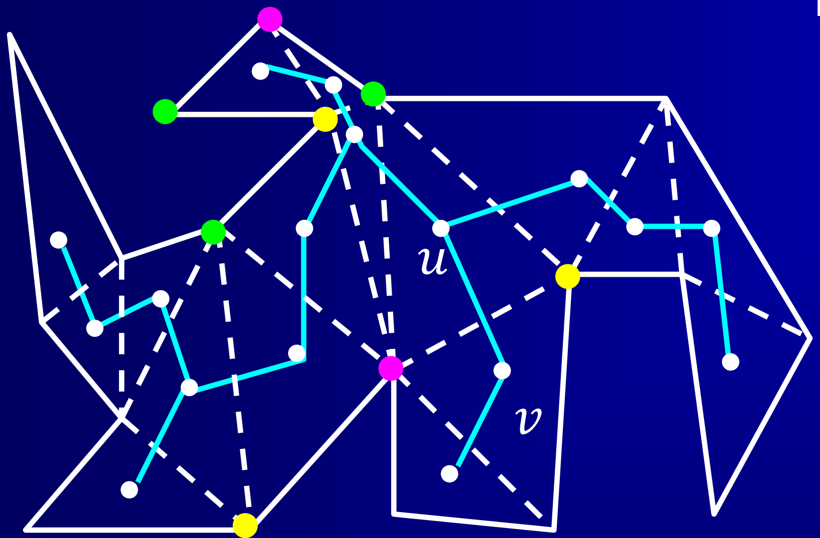
All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

$\Rightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

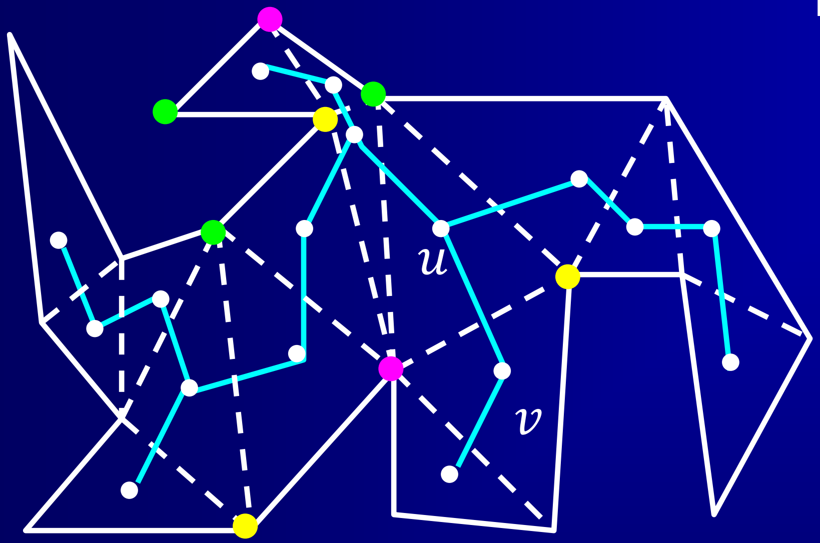
- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

- Suppose node v is visited from u.

$\Rightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

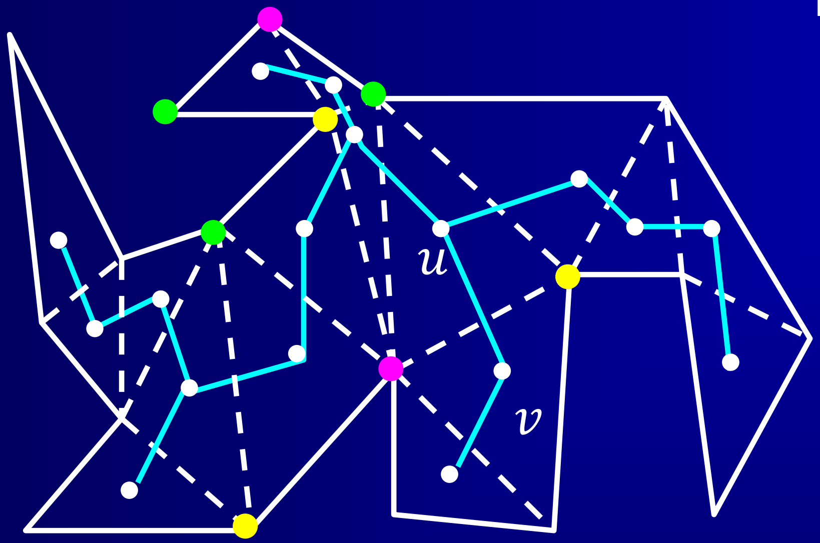
- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

- Suppose node v is visited from u. ⇒ Their triangles $T(v)$ and $T(u)$ are adjacent.

$\Longrightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

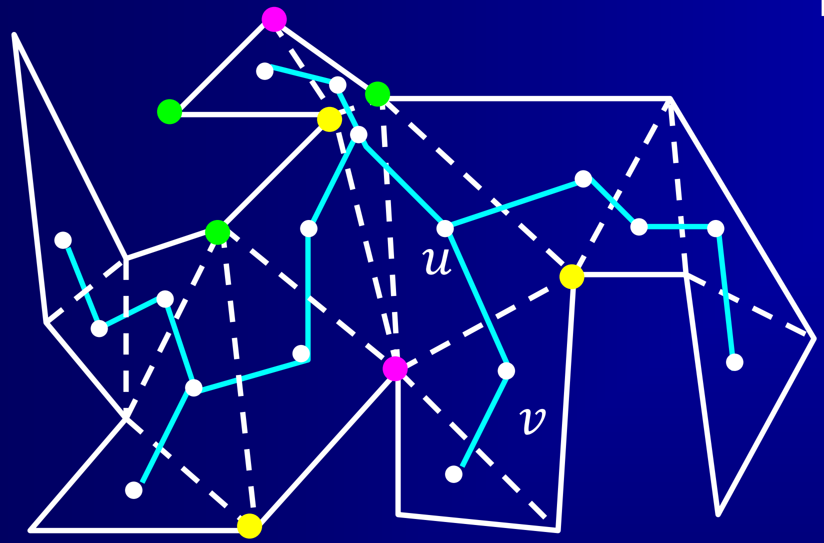
- Suppose node v is visited from u. ⇒ Their triangles $T(v)$ and $T(u)$ are adjacent.

  Only one vertex of $T(v)$ is not colored.

$\Rightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

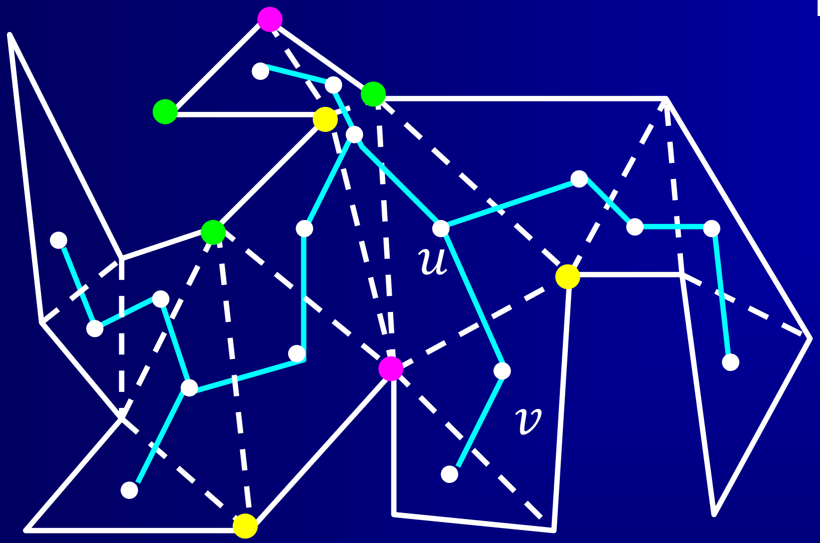
- Suppose node v is visited from u. ⇒ Their triangles $T(v)$ and $T(u)$ are adjacent.

  Only one vertex of $T(v)$ is not colored. ⇒ Its color is uniquely determined.

$\Longrightarrow$

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

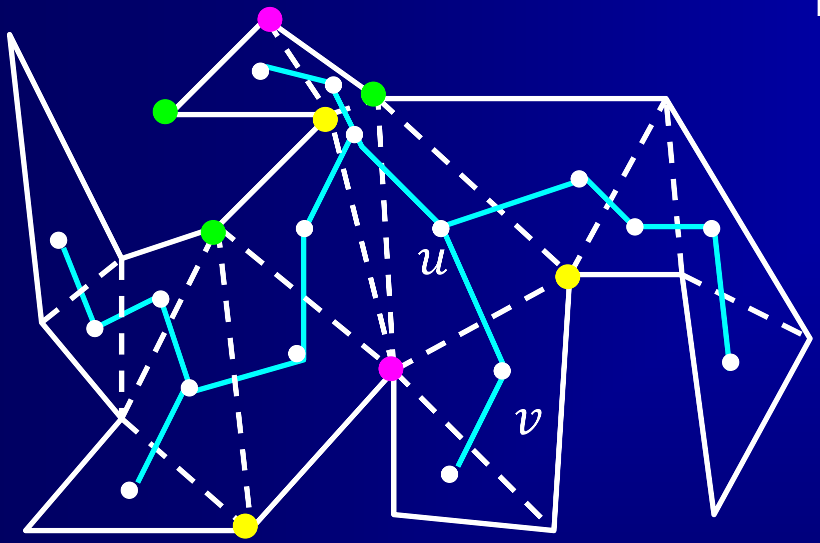
- Suppose node v is visited from u. ⇒ Their triangles $T(v)$ and $T(u)$ are adjacent.

  Only one vertex of $T(v)$ is not colored. ⇒ Its color is uniquely determined.

  Since G is a tree, the other nodes adjacent to v have not been visited yet. Otherwise there exists a cycle (which contradicts that G is a tree.)

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

- Suppose node v is visited from u. ⇨ Their triangles $T(v)$ and $T(u)$ are adjacent.

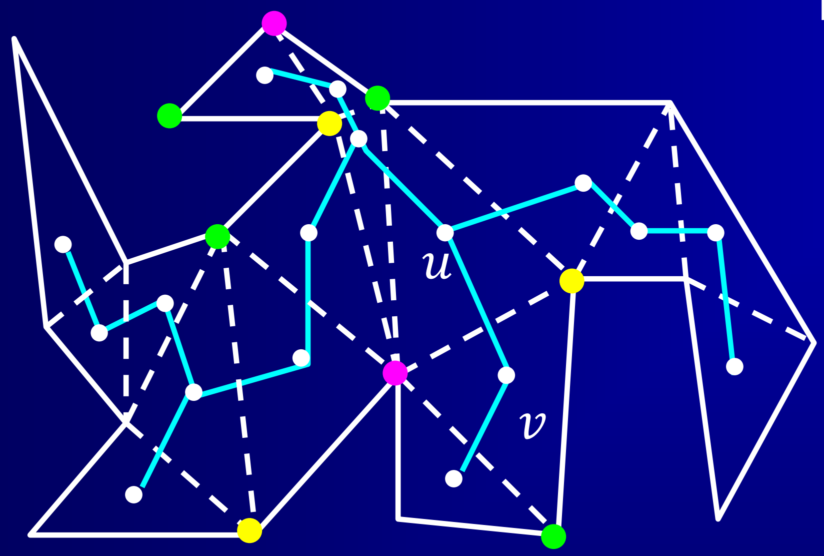
  Only one vertex of $T(v)$ is not colored. ⇨ Its color is uniquely determined.

  Since G is a tree, the other nodes adjacent to v have not been visited yet. Otherwise there exists a cycle (which contradicts that G is a tree.)

  Apply the color to v.

A 3-Coloring Algorithm

A 3-coloring can be found through a graph traversal (such as DFS).



During DFS, maintain the invariant:

All polygon vertices of encountered triangles have been colored such that no adjacent two have the same color.

- Start DFS at any node of G. Color the three vertices of the corresponding triangle.

- Suppose node v is visited from u. ➡ Their triangles $T(v)$ and $T(u)$ are adjacent.

  Only one vertex of $T(v)$ is not colored. ➡ Its color is uniquely determined.

  Since G is a tree, the other nodes adjacent to v have not been visited yet. Otherwise there exists a cycle (which contradicts that G is a tree.)

  Apply the color to v.

A Worst Case

A triangulated polygon can always be 3-colored.

A Worst Case

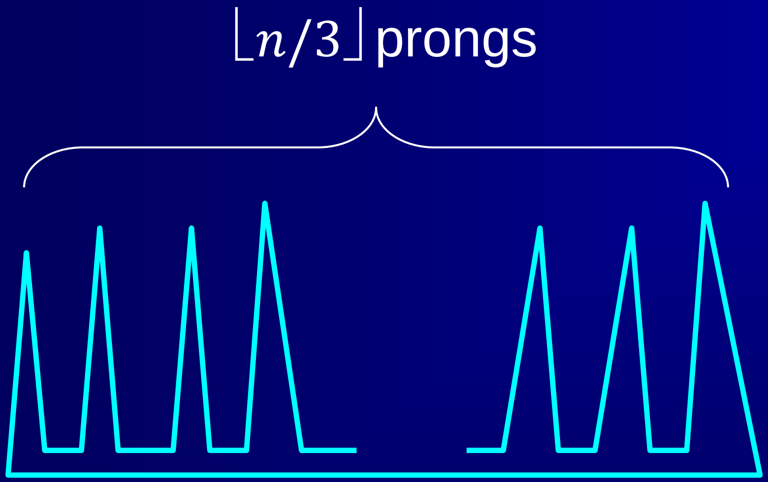
A triangulated polygon can always be 3-colored.

⇨ Any simple polygon can be guarded with $\lfloor n/3 \rfloor$ cameras.

A Worst Case

A triangulated polygon can always be 3-colored.

➡ Any simple polygon can be guarded with $\lfloor n/3 \rfloor$ cameras.

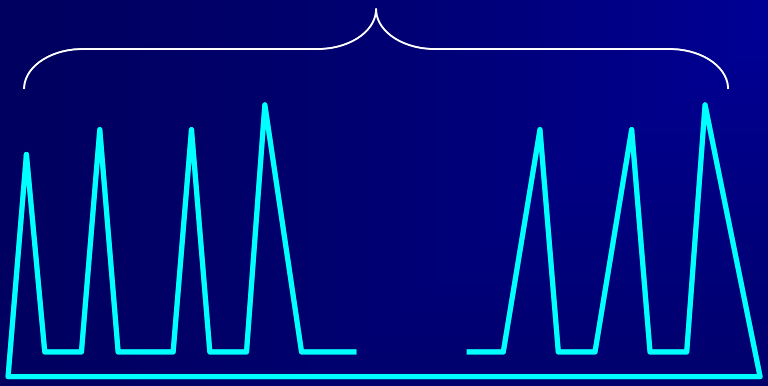


A Worst Case

A triangulated polygon can always be 3-colored.

Any simple polygon can be guarded with $\lfloor n/3 \rfloor$ cameras.

$\lfloor n/3 \rfloor$ prongs



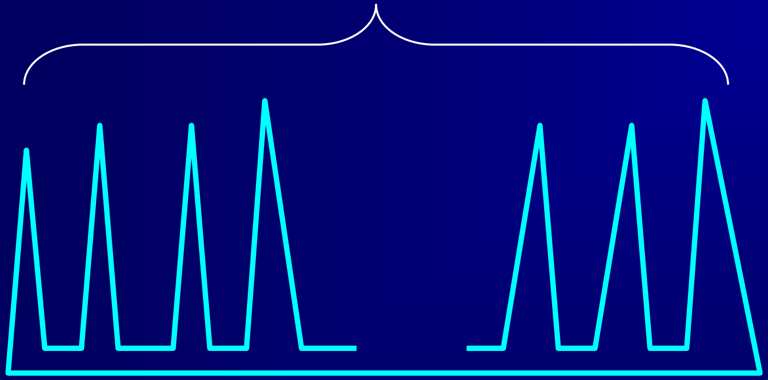
There exists no position at which a camera can oversee two prongs.

A Worst Case

A triangulated polygon can always be 3-colored.

➡ Any simple polygon can be guarded with $\lfloor n/3 \rfloor$ cameras.

$\lfloor n/3 \rfloor$ prongs



There exists no position at which a camera can oversee two prongs.

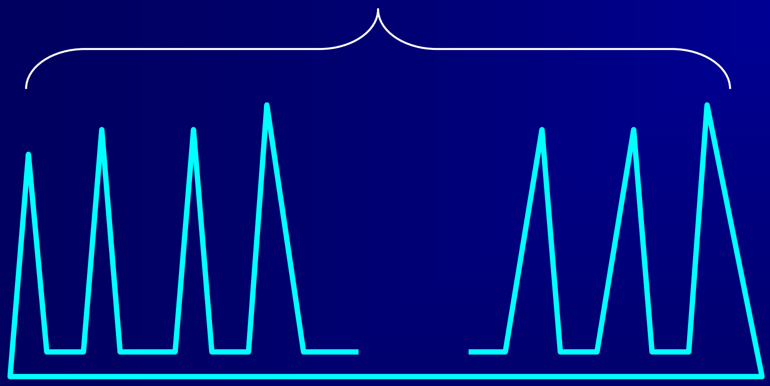
$\lfloor n/3 \rfloor$ cameras are needed.

A Worst Case

A triangulated polygon can always be 3-colored.

➡ Any simple polygon can be guarded with $\lfloor n/3 \rfloor$ cameras.

$\lfloor n/3 \rfloor$ prongs



There exists no position at which a camera can oversee two prongs.

$\lfloor n/3 \rfloor$ cameras are needed.

The 3-coloring approach is optimal in the worst case.

Art Gallery Theorem

For a simple polygon with n vertices, $\lfloor n/3 \rfloor$ cameras are sufficient to have every interior point visible from at least one of the cameras.

Art Gallery Theorem

For a simple polygon with n vertices, $\lfloor n/3 \rfloor$ cameras are sufficient to have every interior point visible from at least one of the cameras.

Solution to the Art Gallery Problem

1. Triangulate a simple polygon with a fast algorithm.

   DCEL representation for the simple polygon so we can visit a neighbor from a triangle in constant time.

2. Generate a 3-coloring by DFS (as presented earlier).

3. Take the smallest color class to place the cameras.